Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning
Main Track

A Credal Least Undefined Stable Semantics for Probabilistic Logic Programs and Probabilistic Argumentation

Victor Hugo Nascimento Rocha , Fabio Gagliardi Cozman
Escola Politécnica, Universidade de São Paulo
{victor.hugo.rocha,fgcozman}@usp.br

Abstract

We present an approach to probabilistic logic programming and probabilistic argumentation that combines elements of the L-stable semantics and the credal semantics. We derive the complexity of inferences, propose an extended version of argumentation graphs with a semantics that maps to the L-stable semantics, and introduce a definition for the probability of an argument.

1 Introduction

In this paper we deal with probabilistic versions both of logic programs and of argumentation frameworks. On the one hand, we propose a semantics for probabilistic logic programs that is motivated by the needs of probabilistic argumentation. On the other hand, we obtain a novel approach to probabilistic argumentation that is derived from the features and semantics of logic programming.

In the remaining sections of the paper, we start with probabilistic logic programming and move to probabilistic argumentation, as this is the easiest way to chain the technical results. However, the work was developed in the reverse direction: we started with a few issues in probabilistic argumentation, then needed a novel approach to probabilistic logic programming. We follow this path in the next paragraph, as it better conveys the motivation for our proposals.

Our intended argumentation scenario is this. Consider an artificial agent that extracts arguments from various sources and organizes the arguments to help and support (not persuade) human agents, possibly dealing with assumptions and probabilities. Work on probabilistic argumentation typically assigns probability directly to abstract arguments, but it is often difficult to interpret the required probabilities. We assume our agent builds probabilistic arguments whose internal structure is based on rules and facts associated with probabilities; from those probabilities the agent can arrive at the probabilities of arguments. The whole pipeline is depicted in Figure 1: rules and probabilities are first extracted and then translated to arguments that support the human user. We do not deal with the extraction process in this paper, but we believe that rules and probabilistic facts offer a better target primary language for extraction than abstract arguments associated with probabilities.

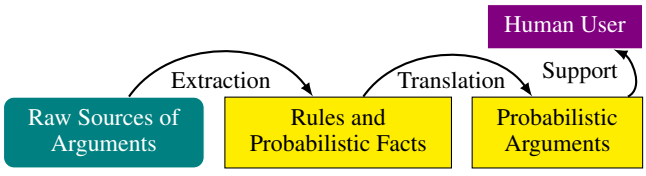


Figure 1: The conceptual pipeline.

In Section 2 we look at probabilistic logic programs that can serve the purpose outlined in the previous paragraph. To do so, we combine two ideas that have been investigated separately: the L-stable semantics for sets of rules and the credal semantics for probabilistic facts. We define the resulting *L-credal* semantics and its basic inference problem. We then derive the complexity of the inference problem. Even though our motivation arises from argumentation theory, we believe that our results are of general interest.

We then present, in Section 3, a new theory of probabilistic argumentation that is based on probabilistic logic programs under the L-credal semantics. We show how arguments can be constructed and how argumentation graphs can be enlarged so that their semantics meshes smoothly with the underlying L-stable semantics. We then propose a definition for the probability of an argument.

Conclusions and plans for future work appear in Section 4.

2 The L-Credal Semantics

Section 2.1 combines well-known concepts, and Section 2.2 introduces the L-credal semantics.

2.1 Rules and Probabilistic Facts

We start with a vocabulary with a finite set of names for *constants* and a finite set of names for *parameterized random variables* (Poole 2003). A parameterized random variable indexes, through its arguments, a set of random variables; for instance, if $\mathsf{brotherOf}$ is a parameterized random variable with two arguments, then $\mathsf{brotherOf}(\mathsf{Mary}, \mathsf{John})$ is a random variable. Each random variable has three values; logic programming typically resorts to labels true, false, and $\mathsf{undefined}$. The underlying sample space is the set of all configurations of random variables.

An *atom* consists of a parameterized random variable with arguments that may be constants or *logical variables*; for instance, $\mathsf{brotherOf}(\mathsf{Mary}, \mathsf{John})$, $\mathsf{brotherOf}(X, \mathsf{John})$,

$\mathsf{brotherOf}(X, Y)$. A *ground atom* is an atom without logical variables (hence, a random variable). An atom can be grounded into a ground atom by replacing its logical variables by constants. The Herbrand base is the (finite) set of all grounded atoms (random variables) that can be built with names in the vocabulary.

A rule is an expression of the form

$$H :- B_1, \ldots, B_m, \mathbf{not}\ B_{m+1}, \ldots, \mathbf{not}\ B_{m+n}\bullet,$$

where H and each B_i is an atom. The atom H is the *head* of the rule; the righthand side is the *body* of the rule. A *fact* is a rule without body, written $H\bullet$. A *logic program* is a set of rules (some of which may be facts).

We could use classic negation $\neg$ and also rules without heads (usually referred to as *constraints*), but, coupled with the semantics discussed later, these constructs can be compiled away (Eiter, Ianni, and Krennwalner 2009).

In the remainder of this paper we assume that all given rules have been grounded with all possible atoms. Thus we are dealing with logic programs without logical variables.

We take uncertainty to be coded by associating probabilities with selected facts as often done in probabilistic logic programming (Fierens et al. 2014; Sato 1995). Such a simple scheme cannot capture every subjective or objective kind of uncertainty in argumentation, but it seems to be an excellent compromise between expressivity and complexity — not just computational complexity, but also the complexity that a human user faces in dealing with the formalism.

We adopt ProbLog's syntax (Fierens et al. 2014) and specify a *probabilistic fact* using the syntax $\alpha :: F\bullet$ to mean, intuitively, that with probability α the fact $F\bullet$ is added to the program (and with probability $1-\alpha$ the fact $F\bullet$ is discarded). A set of rules and probabilistic facts is a *probabilistic logic program*.

Before we move to a discussion of probabilistic semantics, we briefly review a few well-known semantics for logic programs (Caminada et al. 2015). A *three-valued interpretation* of a logic program $\mathbf{P}$ is a pair $(\mathcal{T}, \mathcal{F})$ such that $\mathcal{T} \cap \mathcal{F} = \emptyset$ and both $\mathcal{T}$ and $\mathcal{F}$ contain elements of the Herbrand base of $\mathbf{P}$. A *three-valued model* of $\mathbf{P}$ is an interpretation $(\mathcal{T}, \mathcal{F})$ such that: H is in $\mathcal{T}$ if there is a rule whose head is H and where each B not preceded by $\mathbf{not}$ is in $\mathcal{T}$; H is in $\mathcal{F}$ if every rule whose head is H is such that there is some B not preceded by $\mathbf{not}$ in $\mathcal{F}$. The *reduct* of $\mathbf{P}$ with respect to a three-valued interpretation $\mathcal{I}$, denoted $\mathbf{P}/\mathcal{I}$, is a logic program without $\mathbf{not}$: first, remove from $\mathbf{P}$ every rule that contains $\mathbf{not}\ B$ in its body for some $B \in \mathcal{T}$; then, for each remaining rule, remove $\mathbf{not}\ B$ from the body of the rule if $B \in \mathcal{F}$; finally, replace any remaining occurrences of $\mathbf{not}\ B''$ by some fresh symbol $\clubsuit$ that represents "undefined". Then $\mathbf{P}/\mathcal{I}$ has a unique three-valued model with minimal $\mathcal{T}$ and maximal $\mathcal{F}$ (with respect to set inclusion); denote this model by $\Phi_{\mathbf{P}}(\mathcal{I})$. A *partial stable model* of $\mathbf{P}$ is an interpretation $\mathcal{I}$ such that $\Phi_{\mathbf{P}}(\mathcal{I}) = \mathcal{I}$. The *well-founded model* [*regular*] of $\mathbf{P}$ is the partial stable model with minimal [maximal] $\mathcal{T}$. A *stable model* of $\mathcal{P}$ is a partial stable model without undefined atoms.

2.2 The L-Credal Semantics

Our semantics for probabilistic facts is directly based on Sato's distribution semantics (Sato 1995; Fierens et al. 2014). A *choice* for probabilistic fact $\alpha :: F\bullet$, where F is a ground atom, is a decision as to whether the probabilistic fact is replaced by the fact $F\bullet$ or simply discarded (if F contains variables, we assume a probabilistic per grounding is generated before any semantics is applied). A *total choice* is a set of choices, one per probabilistic fact. All choices are assumed independent; for a set of probability facts $\{\alpha_i :: F_i\bullet\}_{i=1}^N$, the probability of a total choice $\mathcal{O}$ is

$$\mathbb{P}(\mathcal{O}) = \prod_{\substack{(\alpha_i :: F_i\bullet) \text{ not} \\ \text{discarded by } \mathcal{O}}} \alpha_i \prod_{\substack{(\alpha_i :: F_i\bullet) \\ \text{discarded by } \mathcal{O}}} (1-\alpha_i). \quad (1)$$

A total choice transforms a probabilistic logic program into a logic program. Thus, a probability distribution over total choices induces a probability distribution over logic programs. To have a complete semantics, we must decide which semantics to adopt for the induced logic programs.

Existing approaches to probabilistic logic programming adopt either the well-founded or the stable semantics for the logic programs induced by total choices. The well-founded semantics always exists and is unique; when it does not contain undefined values, it is equal to the stable semantics. It is thus not surprising that the well-founded semantics has been proposed in connection with probabilistic facts (Sato, Kameya, and Zhou 2005; Hadjichristodoulou and Warren 2012), for then the unique probability distribution over total choices induces a unique distribution over atoms. However, the well-founded semantics conflates two distinct situations into its undefined values. On the one hand, it may be the case that we *could* accept or reject an atom in more than one way, but the semantics simply refuses to make a decision. To use a well-known example, take the set of rules:

$$G_1 :- \mathbf{not}\ G_2\bullet,\ G_2 :- \mathbf{not}\ G_1\bullet,\ H :- G_1\bullet,\ H :- G_2\bullet.$$

The well-founded semantics leaves all atoms undefined. But we in fact can label G_1 as true and G_2 as false or the other way around while respecting all structural constraints of the logic program. In both cases H gets true, even though it is left undefined by the well-founded semantics. And then there is the entirely different situation, where we *cannot* take neither true nor false. An example is the logic program $\{H :- \mathbf{not}\ H\bullet\}$: atom H cannot be true nor false without breaking some structural assumption about logic programs. Even though the two logic programs in this paragraph are fundamentally different, the well-founded semantics handles both cases through the same undefined value.

The stable model semantics has no place for undefined values, at a cost: there may be, for a logic program, many stable models, or none. For instance, there are two stable models for the first example in the previous paragraph. One assigns false only to G_1 and true to the other atoms; the other assigns false only to G_2 and true to the other atoms. Intuitively, each stable model offers a solution to the constraints imposed by rules. For the second logic program in the previous paragraph, the stable model semantics does not exist, signifying that there is no way to satisfy the constraints.

If a total choice $\mathcal{O}$ leads to a logic program with more than one stable model, we have no information as to how to distribute the probability $\mathbb{P}(\mathcal{O})$. One solution is to distribute that probability mass uniformly over the stable models (Baral, Gelfond, and Rushton 2009; Totis, Kimmig, and Raedt 2021). In that approach, if we have two stable models, each one of them gets $\mathbb{P}(\mathcal{O})\,/2$. One may defend such a strategy through maximum entropy or some other kind of minimum commitment principle (Howson and Urbach 1993). However, it is *not* the case that such solution is neutral when it comes to beliefs and opinions (Walley 1991). A minimum commitment strategy can still be found, as originally proposed by Lukasiewicz (2005); take the set of *all* possible induced distribution over models prescribed by the adopted semantics. For instance, if we have two stable models for total choice $\mathcal{O}$, we can have a distribution that assigns $\mathbb{P}(\mathcal{O})$ to one of them and zero to the other, or $\mathbb{P}(\mathcal{O})\,/2$ to both, and so on. Because sets of probability distributions are often called *credal sets* (Augustin et al. 2014), the resulting semantics has been referred to as the *credal semantics* (Cozman and Mauá 2017).

For each distribution in the semantics, the probabilities of any value of an atom A are obtained by summing probabilities for all models in which A gets the value. Each distribution thus assigns a probability to each total choice $\mathcal{O}$, and of course that probability must agree with Expression 1.

To summarize: the credal semantics of a probabilistic logic program is the set of all probability distributions over stable models of logic programs induced by total choices, such that each probability distribution agrees with the probabilities of total choices. If every total choice induces a single stable model, the credal semantics induces a unique probability distribution over all atoms. In general, the credal semantics induces a closed convex set of probability distributions over atoms (Cozman and Mauá 2017).

Alas, the credal semantics breaks down when some total choices induce logic programs without stable models. In logic programming, where stable models are often the solution to a combinatorial problem (Eiter, Ianni, and Krennwalner 2009), absence of stable models can be useful information: it shows the constraints in the logic program cannot be satisfied. Instead, in probabilistic logic programming, the fact that a few scenarios cannot be satisfied should not break the semantics. This will be particularly important when we later move to probabilistic argumentation: it is not reasonable to stop the analysis of probabilistic arguments just because some scenarios leave a few atoms undefined. We must be able to proceed even when some total choices create scenarios that are partially, or even totally, unsatisfiable.

Our solution will be to adopt the *least undefined semantics*, often referred to as the *L-stable semantics*, for logic programs.

A *least undefined stable model*, or *L-stable model* for short, of a logic program $\mathbf{P}$ is a partial stable model with maximal $\mathcal{T} \cup \mathcal{F}$ (with respect to set inclusion). That is, a L-stable model is a partial stable model with a minimum number of undefined atoms. The set of L-stable models is the L-stable semantics of $\mathbf{P}$ (Sacca 1997). A nice feature of this semantics is that, if a logic program has one or more stable models, those are exactly the L-stable models. However, if a logic program has no stable models, then there will be partial stable models for it, but they will have undefined atoms (not necessarily all atoms will be undefined; just the minimum number of them).

	$v = \mathsf{true}$	$v = \mathsf{false}$	$v = \mathsf{undefined}$
$\mathbb{P}(A = v)$	0.5	0.5	0.0
$\mathbb{P}(B = v)$	0.5	0.5	0.0
$\mathbb{P}(C = v)$	$[0.5, 0.75]$	$[0.25, 0.5]$	0.0
$\mathbb{P}(D = v)$	$[0.5, 0.75]$	$[0.25, 0.5]$	0.0
$\mathbb{P}(E = v)$	0.0	0.5	0.5

	$v = \mathsf{true}$	$v = \mathsf{false}$	$v = \mathsf{inconsistent}$
$\mathbb{P}(A = v)$	0.0	0.5	0.5
$\mathbb{P}(B = v)$	0.25	0.25	0.5
$\mathbb{P}(C = v)$	0.125	0.375	0.5
$\mathbb{P}(D = v)$	0.375	0.125	0.5
$\mathbb{P}(E = v)$	0.0	0.5	0.5

Table 1: Top: tight probability intervals for Example 1; if an interval contains a single number, the interval is replaced by the number. Bottom: probabilities using the SMProbLog approach.

Definition 1. *Given a probabilistic logic program* $\mathbf{P}$*, its* credal least undefined stable *semantics, or* L-credal *semantics, is the set of all probability distributions over L-stable models of programs induced by total choices, such that each distribution agrees with the probabilities of total choices.*

An example should clarify the semantics.

Example 1. *Suppose we have the following rules:*
$\mathsf{worksInTown}(X) :- \mathsf{barber}(X)$**.**,
$\mathsf{worksInRiver}(X) :- \mathsf{fisherman}(X)$**.**,
$\mathsf{worksInTown}(X) :-$ **not** $\mathsf{worksInRiver}(X)$**.**,
$\mathsf{worksInRiver}(X) :-$ **not** $\mathsf{worksInTown}(X)$**.**,
$\mathsf{shaves}(X, Y) :- \mathsf{barber}(X),$ **not** $\mathsf{shaves}(Y, Y)$**.**.
These rules state a few facts about professions in a village, and moreover state that a barber shaves everyone who does not shave himself (this is a combination of well-known examples in logic programming). Suppose additionally that we have two probabilistic facts:
$0.5 :: \mathsf{barber}(\mathsf{John})$**.**, $0.5 :: \mathsf{fisherman}(\mathsf{John})$**.**.
By grounding, we obtain a logic program of the form:
$C :- A$**.**, $D :- B$**.**, $C :-$ **not** D**.**, $D :-$ **not** C**.**, $E :- A,$ **not** E**.**, $0.5 :: A$**.**, $0.5 :: B$**.**. *There are four total choices ($\emptyset$, $\{A\}$, $\{B\}$, $\{A, B\}$), each with probability 0.25. For total choice $\emptyset$, the induced logic program has two L-stable models (both are also stable models): one where every atom is* false *except C that is* true*, and another where every atom is* false *except D that is* true*. For total choice $\{A\}$, there is a single L-stable model (that is also the well-founded model): E is* undefined*, while both A and C are* true *and both B and D are* false*. For total choice $\{B\}$, there is again a single L-stable model (that is the single stable model): A and C and E are* false *and both B and D are* true*. Finally, for total choice $\{A, B\}$ there is a single L-stable model (that is the well-founded model): all atoms are* true *except E that is* undefined*. By adding probabilities across models for all possible distributions in the semantics, we obtain the tight probability intervals in Table 1 (top).* □

Note that the L-credal semantics nicely differentiates between: (i) the uncertainty captured by the probabilities in probabilistic facts; (ii) the uncertainty regarding multiple stable models, captured by probability intervals; (iii) the uncertainty arising from inability to satisfy constraints, leading to non-zero probability for undefined values.

Previous literature has discussed situations where total choices induce logic programs without stable models. One proposal is to automatically repair logic programs whose rules cannot be satisfied (Ceylan, Lukasiewicz, and Peñaloza 2016). Another proposal in the literature is to set all atoms to a value inconsistent whenever a stable model does not exist (Totis, Kimmig, and Raedt 2021). As an interesting comparison, we collect in Table 1 (bottom) the probabilities obtained via the SMProbLog approach (Totis, Kimmig, and Raedt 2021), where uniform probabilities are distributed over multiple stable models and absence of stable models puts probability mass into inconsistent values. Note how large are the probabilities of inconsistent values, even for atoms that can always be assigned other values.

As a final point, we argue that words such as true, false and undefined are not really appropriate here. It seems better to use unsatisfiable or undecided rather than undefined, to avoid weird sentences such as "the probability that X is undefined" or, worse, "the conditional probability that X is undefined given that Y is undefined" (Cozman and Mauá 2017). And it seems also better to use accepted rather than true and rejected rather than false, to avoid controversies related to mixtures of probabilities and three-valued logic. From now on we use accepted, rejected, and undecided.

2.3 The Complexity of Inferences

The basic inference problem in probabilistic logic programming is to compute the marginal or conditional probability of a ground atom given a probabilistic logic program. For instance, one may be interested in the probability that John is not a barber given that he works in the river: in Example 1, this probability value can be anywhere within the probability interval $[1/3, 1/2]$. We refer to $\underline{\mathbb{P}}(H) \doteq \inf \mathbb{P}(H)$ as the *lower* probability of atom H; similarly, to $\overline{\mathbb{P}}(H) \doteq \inf \mathbb{P}(H)$ as the *upper* probability of atom H (Augustin et al. 2014; Walley 1991). And we refer to $\underline{\mathbb{P}}(H|G) \doteq \inf_{\mathbb{P}:\mathbb{P}(G)>0} \mathbb{P}(H|G)$ as the *lower conditional* probability of H given G; the latter is only defined if $\overline{\mathbb{P}}(G) > 0$ (likewise, $\overline{\mathbb{P}}(H|G) \doteq \sup_{\mathbb{P}:\mathbb{P}(G)>0} \mathbb{P}(H|G)$ is the upper conditional probability).

We must define a precise decision problem in order to study the complexity of inferences. Our *inference problem* receives: a grounded logic program $\mathbf{P}$ with probabilistic facts all defined by rational numbers, a nonempty set of *query atoms* that belong to the logic program, a set of selected ground atoms that belong to the logic program, referred to as the *evidence*, and a rational number γ. Denote by $\mathcal{Q}$ the event that the query atoms are all accepted, and by $\mathcal{E}$ the event that the evidence atoms are all accepted. If the evidence is empty, then the output is YES when $\underline{\mathbb{P}}(\mathcal{Q}) > \gamma$. If the evidence is nonempty and $\overline{\mathbb{P}}(\mathcal{E}) = 0$, then the output is NO (that is, "no probability measure satisfies $\mathbb{P}(\mathcal{Q}|\mathcal{E}) > \gamma$"). And if $\overline{\mathbb{P}}(\mathcal{E}) > 0$, then the output is YES if $\underline{\mathbb{P}}(\mathcal{Q}|\mathcal{E}) > \gamma$ and NO otherwise.

We will show that this inference problem is complete for a complexity class in Wagner's Polynomial Counting Hierarchy (Wagner 1986). We take as known concepts such as complexity classes, decision problems, many-one polynomial reductions, oracles; detailed definitions can be found elsewhere (Papadimitriou 1994). We use well-known complexity classes $\mathtt{P}$, $\mathtt{NP}$, $\mathtt{coNP}$. We also use the complexity class $\mathtt{PP}$: a language $\mathcal{L}$ is in $\mathtt{PP}$ when there is a nondeterministic Turing machine $\mathbf{M}$ such that $l \in \mathcal{L}$ iff more than half of computation paths of $\mathbf{M}$ accept l. The Polynomial Counting Hierarchy is the collection of complexity classes that includes $\mathtt{P}$ such that if $\mathcal{C}$ is in the hierarchy then so are the classes of decision problems computed by oracle machines $\mathtt{PP}^{\mathcal{C}}$, $\mathtt{NP}^{\mathcal{C}}$ and $\mathtt{coNP}^{\mathcal{C}}$. The Polynomial Counting Hierarchy therefore contains the Polynomial Hierarchy, which includes classes such as $\Sigma_k^p = \mathtt{NP}^{\Sigma_{k-1}^p}$ and $\Pi_k^p = \mathtt{coNP}^{\Sigma_{k-1}^p}$ for $k > 0$, with $\Sigma_0^p = \Pi_0^p = \mathtt{P}$, and also counting classes with oracles in the polynomial hierarchy, such as $\mathtt{PP}^{\Sigma_k^p}$.

We can now state our result:

Theorem 1. *The inference problem is* $\mathtt{PP}^{\Sigma_2^p}$*-complete.*

To prove Theorem 1 on the complexity of the inference problem, we need to adapt some previous results about the credal semantics. First recall that an infinitely monotone Choquet capacity is a set-function c from sets in an algebra over a set Ω to real numbers in $[0, 1]$ such that $c(\Omega) = 1 - c(\emptyset) = 1$ and $c(\cup_{i=1}^n E_i) \geq \sum_{J \subseteq \{1,\ldots,n\}} (-1)^{|J|-1} c(\cap_{j \in J} E_j)$ for any sets of events E_i in the algebra (Augustin et al. 2014). A set of functions dominates a Choquet capacity c if every function in the set is uniformly larger or equal to c. We then have:

Theorem 2. *The L-credal semantics of a logic program is a closed convex set of probability distributions that is the largest set dominating an infinitely monotone Choquet capacity.*

Proof. Identical to the proof of Theorem 1 of (Cozman and Mauá 2017). □

The proof of our complexity result will also need the following consequence of Theorem 2:

$$\underline{\mathbb{P}}(\mathcal{Q}|\mathcal{E}) = \frac{\underline{\mathbb{P}}(\mathcal{Q} \cap \mathcal{E})}{\underline{\mathbb{P}}(\mathcal{Q} \cap \mathcal{E}) + \overline{\mathbb{P}}(\mathcal{Q}^c \cap \mathcal{E})} \tag{2}$$

provided the denominator is larger than zero (Molchanov 2005). If the denominator of the latter expression is zero, then $\underline{\mathbb{P}}(\mathcal{Q}|\mathcal{E})$ is equal to one if there is *any* distribution such that $\mathbb{P}(\mathcal{Q} \cap \mathcal{E}) > 0$, and is undefined otherwise.

The proof of membership of Theorem 1 depends a few additional facts based on Theorem 2. To compute a lower probability $\underline{\mathbb{P}}(\mathcal{Q})$, we can go over all total choices and, for each total choice, examine whether all L-stable models are such that $\mathcal{Q}$ is obtained; if that happens, we add the probability of that total choice to some accumulator. At the end, the accumulator (that started from zero) contains the desired lower probability. That is, to obtain a lower probability we must

go over the total choices running *cautious (logical) inference* for each logic program (sometimes referred to as *skeptical* inference) (Eiter, Leone, and Saccá 1998). Similarly, to obtain an upper probability we must run *brave (logical) inference* for each induced logic program – that is, we must check whether $\mathcal{Q}$ is obtained in some L-stable model (sometimes referred to as *credulous* inference).

We can now present the proof of Theorem 1:

Proof. To prove membership in $\text{PP}^{\Sigma_2^p}$, we piece together some parts of previous proofs in the literature. First, we use the first part of the proof of Theorem 16 by Cozman and Mauá (2017) to build a nondeterministic Turing machine that generates a total choice (with probability given by Expression (1)). Then, for a total choice $\mathcal{T}$ selected through a polynomial sequence of non-deterministic moves, the machine builds the induced non-probabilistic logic program $\mathbf{P}_{\mathcal{T}}$. As in that proof, the Turing machine branches into different paths so as to decide whether $k_1\underline{\mathbb{P}}(\mathcal{Q} \cap \mathcal{E}) > k_2\overline{\mathbb{P}}(\mathcal{Q}^c \cap \mathcal{E})$ for suitable constants k_1 and k_2 (this inequality is derived from Expression (2)). Some of the paths require cautions inference, while others require brave inference; the specific branches are detailed in the proof of Theorem 25 by Cozman and Mauá (2017). Now, cautions inference can be solved by calling an oracle in Π_2^p. To see this, consider the decision as to whether an atom belongs to all L-stable models. It is enough to generate a L-stable model that does not contain the atom and negate the output. To do so, first generate, using a polynomial time nondeterministic machine, an interpretation $\mathcal{I}$; a polynomial time nondeterministic machine can be used as oracle to decide whether $\mathcal{I}$ is in fact a L-stable model. A similar reasoning shows that brave inference can be solved by calling an oracle in Σ_2^p.

To prove hardness, we start with the following $\text{PP}^{\Sigma_2^p}$-complete problem (Wagner 1986): Decide whether

$$\#X_1, \ldots, X_n : \forall Y_1, \ldots, Y_m : \exists Z_1, \ldots, Z_s : \phi > M,$$

where ϕ is a propositional formula in all propositional symbols bounded to quantifiers and in Conjunctive Normal Form (CNF) where each clause has at most three literals (propositional variables or propositional variables preceded by negation), and where M is an integer. The symbol $\#$ denotes a *counting* quantifier whose result is to count the number of configurations of bounded propositional variables that satisfy the enclosed formula.

Start with the Σ_2^p-complete decision problem (Papadimitriou 1994): decide whether Θ holds, where $\Theta \doteq \forall Y_1, \ldots, Y_m : \exists Z_1, \ldots, Z_s : \phi$. We treat ϕ as a set of clauses and each clause c as a set of literals. This decision problem can be reduced to the computation of the semi-stable semantics of an argumentation framework (Dvořák 2012, Theorem 12); an atom ϕ is introduced so that formula ϕ holds iff atom ϕ is accepted in all semi-stable argumentation labelings of the argumentation framework. Now we could translate this argumentation framework into a logic program using a well-known transformation (Caminada et al. 2015);[1] this translation has the important property that each labeling of the original argumentation framework corresponds to a L-stable model of the resulting program and vice-versa (Caminada et al. 2015, Theorem 29). Now consider including into this construction the counting quantifier and associated propositional variables X_i. First, we have a formula ϕ containing clauses where some literals may mention the X_i. Include arguments X_i and $\overline{X}_i$ in the argumentation framework above, and treat literals with X_i as all other propositional variables in Dvořák's reduction. The next step is to translate the argumentation framework into a logic program and replace, for each pair X_i and $\overline{X}_i$, the rules with them as heads by the probabilistic fact $0.5 :: X_{i\bullet}$ and the rule $\overline{X}_i :- X_{i\bullet}$. The computation of the lower probability of the atom ϕ yields the probability of the total choices for which ϕ is in all L-stable models; using Dvořák's results, that is the probability ρ that formula Θ holds. By comparing $\rho > 1/2$, we decide whether $\#X_1, \ldots, X_n : \Theta > 2^{n+m+s-1}$. □

[1] This translation is discussed in Section 3.3 (footnote 2).

Note that the corresponding result for the credal *stable* semantics yields $\text{PP}^{\Sigma_1^p}$-completeness (Cozman and Mauá 2017); the L-stable semantics moves us one oracle up.

3 Probabilistic Argumentation: A Proposal

As we mentioned in Section 1, we began this work by looking for a formalism to represent arguments with uncertainty. In this section we offer an approach that, we feel, fixes a few difficulties found in existing approaches. In Section 3.1 we review terminology and existing literature on (probabilistic) argumentation frameworks. Section 3.2 describes our proposal, and Section 3.3 introduces a novel extension of argumentation graphs that is needed to connect logic programs and sets of arguments. And Section 3.4 introduces a new definition for the probability of an argument.

3.1 A Bit of Background on (Probabilistic) Argumentation Frameworks

Argumentation theory has been studied for centuries (van Eemeren et al. 2014); within artificial intelligence, two strong research agendas have been pursued: Abstract Argumentation Frameworks (AAF) and Assumption-based Argumentation Frameworks (ABAF). They have many variants; due to space constraints, we stick to basic definitions.

An AAF is a pair $(\mathcal{A}, \mathcal{R})$ where $\mathcal{A}$ is a finite set of arguments and $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ is a relation between arguments (Dung 1995). If $(A, B) \in \mathcal{R}$, we say that A *attacks* B. An *argumentation labeling* associates each argument with a label, either accepted, rejected, or undecided. A *complete argumentation labeling* L is an argumentation labeling such that: (i) if argument A is rejected then there is argument B that is accepted and that attacks A; (ii) if argument A is accepted then for all arguments B that attack A we must have that B is rejected; (iii) if argument A is undecided then there is no argument B that is accepted and attacks A, and there is an argument C that attacks A and is not rejected. A *grounded argumentation labeling* is a complete argumentation labeling where the set of accepted arguments is minimal. A *preferred argumentation labeling* is a complete argumentation labeling where the set of accepted arguments

is maximal. A *stable argumentation labeling* is a complete argumentation labeling with no undecided argument. A *semi-stable argumentation labeling* is a complete argumentation labeling where the set of undecided arguments is minimal. A semantics prescribes the labels for all arguments: the complete/grounded/preferred/stable/semi-stable semantics prescribe the corresponding argumentation labelings. An *argumentation graph* associated with an AAF is a graph where each node is an argument and each directed arrow corresponds to an attack.

An ABAF on the other hand emphasizes the structure of arguments (Bondarenko et al. 1997). Usually such a framework consists of a language, a set of rules, a set of *assumptions* in the language, and a *contrary* function that indicates a sentence that defeats an assumption (Toni 2014). An argument can be viewed as a finite tree with an element of the language in the root (the *conclusion*), nodes that represent rules, and assumptions in the leaves (Caminada and Schulz 2017). Logic programs and ABAFs can be directly translated to one another (Caminada and Schulz 2017). This sort of translation to/from logic programming applies also to many other structured argumentation and non-monotonic formalisms (Heyninck and Strasser 2016; Heyninck 2019). Given this, we stay with logic programs as the main structured representation for argumentation schemes, even though we certainly agree that many features such as supports, weights and preferences (Modgil and Prakken 2014), cannot be captured by logic programs.

Interest in probabilistic versions of argumentation frameworks has been growing for some time. Initial efforts focused on structured argumentation, some of them closer to probabilistic logic (Haenni 2001; Haenni 2009), while others concentrating on probabilistic ABAFs (Dung and Thang 2010; Riveret et al. 2007); however, their assumptions and probabilistic assessments are significantly different than ours. Recent work (Čyras, Heinrich, and Toni 2021; Hung 2017) operates closer to us by combining ABAFs with (what amounts to) probabilistic facts, but with semantics that are not related to our L-credal semantics. Relatively less work has appeared on probabilistic ABAFs than on probabilistic AAFs, and most (almost all) proposals for probabilistic argumentation assume that probabilities are assigned to arguments or to attacks, with differences on how these probabilities are to be interpreted. Fortunately a detailed recent survey can be consulted (Hunter et al. 2021), so here we only offer a very brief review of probabilistic argumentation and skip important topics that we do not need in this paper.

Probabilistic AAFs can basically be of two kinds. In the *constellation* approach, uncertainty lies in the graph topology: arguments and attacks may be present with given probabilities (Li, Oren, and Norman 2011; Mantadelis and Bistarelli 2020). In the *epistemic* approach the graph topology is fixed and there is uncertainty as to which degree arguments are believed — or justified, or acceptable; the interpretation varies in the literature (Baroni, Giacomin, and Vicig 2014; Hunter, Polberg, and Thimm 2020; Thimm 2012).

The constellation approach usually relies on the concepts of acceptance and rejection from (non-probabilistic) AAFs; for instance, one may be interested in the probability that some atom gets some argumentation labeling. The epistemic approach typically offers more notions of acceptance or conditions of rational behavior, often based on probabilistic ideas (for instance, an argument may be accepted only if its probability is larger than some threshold). In that sense, the epistemic approach moves away from AAFs in an attempt to model the human agents involved in argumentation.

As noted by Prakken (2018), in the constellation approach the probabilities are extrinsic to arguments, as they carry uncertainty about arguing agents, while in the epistemic approach probabilities are intrinsic to arguments. Clearly, one can contemplate mixtures of both approaches (Riveret et al. 2018). An interesting mixture appears in the work by Totis, Kimmig, and Raedt (2021): they employ probabilistic logic programming to build arguments but still assume that probabilities are assigned to arguments and attacks.

3.2 Building Probabilistic Arguments

A challenge in probabilistic argumentation is the lack of consensus over the meaning of probabilities (Prakken 2018). Existing proposals typically start from the probability that an argument is accepted, or that it is justified, or even that it is true; often it feels these approaches assume probabilities over events that should actually be the end result of argumentation. Another challenge is that almost all approaches are based on probabilities over supposedly independent abstract arguments; even approaches related to ABAFs often assign probabilities to arguments, not to their components. This strategy is not satisfactory when arguments must be extracted from human discourse (Figure 1), as in real life many arguments depend on common uncertain sources. The structure of arguments must matter.

Given this, we believe that probabilistic argumentation should be assumption-based, with probabilities attached to constituent atoms. Because logic programs capture most assumption-based argumentation schemes, we take that probabilistic logic programming is the right language from which to build arguments.

However, we also think there is significant value in argumentation graphs when it comes to interacting with human users. One advantage is that the user can focus on subsets of a logic program (the arguments) and look for explanations and justifications there. Another advantage is that the user can be informed about the probability of arguments and not just of atoms, thus enlarging the scope of the interaction.

Fortunately, there are well-known translations from logic programs to argumentation frameworks. We adopt the *WCG* (Wu, Caminada and Gabbay) translation algorithm (Caminada et al. 2015). We believe this algorithm is the best one at capturing the notion of an argument.

We summarize the WCG algorithm as follows. Starting with a set of rules, process one rule at a time. If we find a rule of the form H :− $\mathbf{not}\ B_1, \ldots, \mathbf{not}\ B_n$., then we generate an argument A written as $\langle H \leftarrow \mathbf{not}\ B_1, \ldots, \mathbf{not}\ B_n \rangle$. We say A has *conclusion* H, *rules* $\{H :- \mathbf{not}\ B_1, \ldots, \mathbf{not}\ B_n.\}$, *vulnerabilities* $\{B_1, \ldots, B_n\}$, and a singleton set of *subarguments* that contains A itself. Suppose instead we find a rule of the form $H :- B_1, \ldots, B_m, \mathbf{not}\ B_{m+1}, \ldots, \mathbf{not}\ B_{m+n}$.

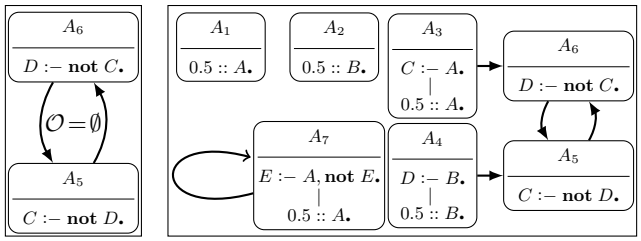


Figure 2: Arguments and graphs generated by running the WCG algorithm for a total choice (left) and with probabilistic facts (right).

Refer to this rule as r and suppose further that, for each $B_1, \ldots, B_m$ there is an argument A_i with conclusion B_i such that r is not in the set of rules of A_i. Then r generates an argument A written as $\langle H \leftarrow \langle A_1 \rangle, \ldots, \langle A_m \rangle, \mathbf{not}\ B_{m+1}, \ldots, \mathbf{not}\ B_{m+n} \rangle$. We say A has *conclusion* H, *rules* consisting of the union of the sets of rules for each A_i and the singleton with rule r, *vulnerabilities* consisting of the union of the sets of vulnerabilities for each A_i and the set $\{B_{m+1}, \ldots, B_{m+n}\}$, *subarguments* consisting of the union of the subarguments for each A_i and the singleton containing just A itself. Intuitively, an argument can be drawn as a tree-like structure, with the root containing a top rule whose head is the conclusion, and additional rules whose conclusions belong to the right hand side the top rule, and so on recursively. Say that A *attacks* A' iff the conclusion of A belongs to vulnerabilities of A'. Nicely, given a set of rules, this procedure generates an AAF.

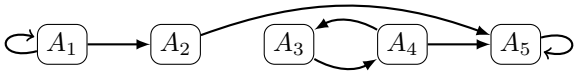
Hence, a probabilistic logic program generates, for each total choice $\mathcal{O}$, an AAF through the WCG algorithm. The set of AAFs and their probabilities can be drawn in compact form if one is willing to depict the structure of arguments:

Example 2. *Consider again Example 1. Figure 2 (left) depicts the argumentation graph obtained by selecting total choice $\mathcal{O} = \emptyset$ and running the WCG algorithm. The argumentation graph show the structure of the arguments, even though this is not usually drawn in an argumentation graph. Figure 2 (right) is a bit different; there the arguments are built running the WCG algorithm* as if *the probabilistic facts were facts, and leaving them with their probabilities. On the one hand, if we erase the probabilities, we have the argumentation graph obtained by selecting total choice $\mathcal{O} = \{A, B\}$ and running the WCG algorithm. On the other hand, if we leave the probabilities, the right graph captures the whole gamut of total choices and their probabilities.* □

In Section 3.4 it will be important to use probabilistic logic programs "inside" arguments as generated in this example (we will refer to them as *attached probabilistic logic programs*).

At this point we have a complete pipeline corresponding to Figure 1. Alas, there is an unsatisfying snag. Existing results show that it is *impossible* to find a semantics for AAFs that maps back to the L-stable semantics (Caminada et al. 2015, Theorem 24). The last sentence must be made more precise. Suppose that we start with a logic program $\mathbf{P}$ and we generate an AAF by running the WCG algorithm. Suppose we find an argumentation labeling for arguments using a selected argumentation semantics. Suppose further that, for each atom C, we find the highest label amongst the labels of arguments whose conclusion is C, where accepted $>$ undecided $>$ rejected. We then assign respectively accepted/undecided/rejected to the atom C if the highest argument label is accepted/undecided/rejected. Can this *back mapping* procedure always generate L-stable models for the original program $\mathbf{P}$? Unfortunately, no, at least not with the tools we have so far.

Example 3. *Here is a short version of the key example by Caminada et al. (2015). Consider the following rules: $E :- \mathbf{not}\ E.$, $A :- \mathbf{not}\ B.$, $B :- \mathbf{not}\ A.$. Now build a logic program $\mathbf{P}_1$ consisting of these rules and $C :- \mathbf{not}\ E.$ and $C :- \mathbf{not}\ A, \mathbf{not}\ C.$. And build another logic program $\mathbf{P}_2$ consisting of the three previous rules and $D :- \mathbf{not}\ E.$ and $C :- \mathbf{not}\ A, \mathbf{not}\ C, \mathbf{not}\ D.$. The WCG algorithm generates the same argumentation graph for both logic programs:*



Programs $\mathbf{P}_1$ and $\mathbf{P}_2$ have respectively two and one L-stable models. Alas, it is not possible for a single graph, whatever the argumentation semantics, to map back to two distinct models. □

Somewhat surprisingly, the back mapping procedure *does* work for several pairs of program/argumentation semantics: We obtain respectively a partial stable/well-founded/regular/stable model of $\mathbf{P}$ if we use complete/grounded/preferred/stable argumentation labelings. But no argumentation semantics can always back map into the L-stable semantics.

Consequently, argumentation graphs are not really appropriate to convey the L-stable semantics, and hence the L-credal semantics, to a human user as envisioned in Figure 1. The next section solves this problem.

3.3 Conclusion-augmented Argumentation Frameworks (CAFs)

We wish to keep the L-stable semantics for logic programs but still be able to recover L-stable models from argumentation graphs. Our way out is straightforward: enlarge AAFs so that each argument is explicitly associated with a conclusion. As argumentation is about conclusions, it makes sense to give the latter a more prominent status. And the resulting framework is still rather abstract as no argument exhibits internal structure.

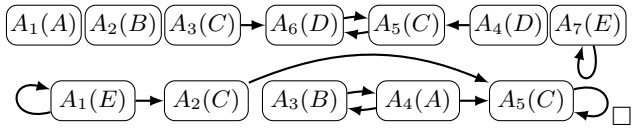
In fact, the same idea has been proposed in connection with *Claim-Augmented Argumentation Frameworks* (Dvorák and Woltran 2019; Dvořák, Rapberger, and Woltran 2020; Rapberger 2020). We prefer to use the term *conclusion* instead of *claim*, as the former seems to better fit the previous literature we work upon (Caminada et al. 2015). We also develop here what we hope is a more direct route to obtain correspondences to logic programming. Despite this, both frameworks are equivalent so the proposal itself is not a novel contribution of the present paper.

We define:

Definition 2. *A Conclusion-augmented Argumentation Framework (CAF) is a tuple $(\mathcal{A}, \mathcal{S}, f, \mathcal{R})$, where $\mathcal{A}$ is a finite set of arguments, $\mathcal{S}$ is a set of atoms referred to as conclusions, f is a function from arguments to conclusions, and $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ is an attack relation as before.*

We write that for a conclusion c, the set of arguments associated with it is denoted by $f^{-1}(c)$. An *conclusion-augmented argumentation graph* is a directed graph where each node contains an argument and its conclusion, and each arrow goes from a node containing an attacking argument to a node containing an attacked argument.

Example 4. *The program in Example 2 with total choice $\mathcal{O} = \{A, B\}$ and program $\mathbf{P}_1$ in Example 3 respectively lead to the following conclusion-augmented argumentation graphs:*



Given a CAF, we can label each conclusion as accepted, undecided, or rejected. Clearly such a *conclusion labeling* maps to an argumentation labeling through the function f. And the previous back-mapping procedure produces a conclusion labeling out of any given argumentation labeling.

Obviously, a CAF can be generated by any logic program by running the WCG algorithm with the small change that arguments are explicitly associated with their conclusions at the end. We refer to the resulting algorithms as the *extended* WCG algorithm.

Definitions related to argumentation labelings can be readily adapted to conclusion labelings. For instance, consider the definition of complete conclusion labeling:

Definition 3. *Let $(\mathcal{A}, \mathcal{S}, f, \mathcal{R})$ be a CAF and $\mathcal{L}$ a conclusion labeling of $\mathcal{S}$. Then $\mathcal{L}$ is a* complete conclusion labeling *iff (i) if a conclusion a is* rejected *then for every argument $A \in f^{-1}(a)$ there is an argument B that attacks A with* accepted *conclusion $f(B)$; (ii) if a conclusion a is* accepted *then for some argument $A \in f^{-1}(a)$, each argument B that attacks A must have a* rejected *conclusion $f(B)$. (iii) if a conclusion a is* undecided *then there is no argument $A \in f^{-1}(a)$ for which all arguments B that attack A have $f(B)$ as* rejected*; and for some argument $A \in f^{-1}(a)$, there is no argument B that attacks A with* accepted *conclusion $f(B)$, and there is an argument C that attacks that same A and whose conclusion $f(C)$ is not* rejected.

Using this definition, other semantics immediately follow:

Definition 4. *Let L be a conclusion labeling. L is grounded iff it is complete and the set of* accepted *conclusions is minimal. L is preferred iff it is complete and the set of* accepted *conclusions is maximal. L is stable iff it is complete and no conclusion is left* undecided*. L is semi-stable iff it is complete and the set of* undecided *conclusions is minimal.*

We then obtain the desired correspondence between logic program and argumentation semantics:

Theorem 3. *Let $\mathbf{P}$ be a logic program and $\mathbf{C} = (\mathcal{A}, \mathcal{S}, f, \mathcal{R})$ the CAF generated by the extended WCG algorithm. Then the complete, preferred, grounded, stable and semi-stable conclusion labelings of $\mathbf{C}$ are identical to the labels assigned respectively by the partial stable, regular, well-founded, stable and L-stable models of $\mathbf{P}$.*

Proof. As proved by Caminada et al. (2015), the complete semantics for $\mathbf{C}$ yields the same results as the partial stable semantics applied to $\mathbf{P}$. As the other semantics are computed by maximizing/minimizing specific labels at the conclusion level, they are equivalent to applying the desired semantics directly in the logic program. □

To conclude this section, we ask: Is it possible to translate *any* CAF to a logic program with equivalent semantics over conclusions? There is a well-known procedure that generates a logic program from an AAF such that all argumentation semantics discussed here are preserved, except the semi-stable semantics (Caminada et al. 2015).[2] Can a similar result be found for CAFs? In fact, we can do even better with CAFs. Consider the following procedure:

Definition 5. *Let $\mathbf{C} = (\mathcal{A}, \mathcal{S}, f, \mathcal{R})$ be a CAF. For each argument A, generate the rule $f(A) :- \mathbf{not}\ f(B_1), ..., \mathbf{not}\ f(B_m)$**.** *where the B_i are the arguments that attack A. Denote by $\mathbf{P_C}$ the logic program consisting of these rules.*

The next theorem proves the desired equivalence between the semantics before and after the translation.

Theorem 4. *Let $\mathbf{C} = (\mathcal{A}, \mathcal{S}, f, \mathcal{R})$ be a CAF and $\mathbf{P_C}$ be the corresponding logic program produced by Definition 5. Then the partial stable, regular, well-founded, stable, L-stable models of $\mathbf{P_C}$ assign the same labels to atoms as labels assigned respectively by the complete, preferred, grounded, stable, semi-stable conclusion labelings of $\mathbf{C}$.*

Proof. Denote by $\mathbf{C_{P_C}}$ the CAF produced by the extended WCG algorithm with $\mathbf{P_C}$ as input. Theorem 3 shows that labels are the same for $\mathbf{P_C}$ and $\mathbf{C_{P_C}}$ for all semantics in the theorem. Hence, if $\mathbf{C}$ and $\mathbf{C_{P_C}}$ produce the same conclusion labeling, we are done. Note that we allow $\mathbf{C}$ to have more than one argument with the same conclusion and that is attacked by the same arguments while $\mathbf{C_{P_C}}$ can have just one "copy" of that argument. Such repeated arguments in $\mathbf{C}$ do not affect the conclusion labelings. Now, if we assume that $\mathbf{C}$ and $\mathbf{C_{P_C}}$ have distinct conclusion labelings, the set of conclusions from their arguments is different and/or the set of vulnerabilities associated with them is different. In the translation shown in Definition 5, each rule of $\mathbf{P_C}$ contains in its head H the conclusion associated with an argument from $\mathbf{C}$ and, from extended the WCG algorithm, we know each rule becomes an argument in $\mathbf{C_P}$. Therefore the set of conclusions is the same. Similarly, the set of vulnerabilities must be the same in $\mathbf{C}$ and $\mathbf{C_{P_C}}$. Because of that, $\mathbf{C}$ and $\mathbf{C_{P_C}}$ produce the same conclusion labelings. □

[2] This is the translation also mentioned in the proof of Theorem 1: each argument maps to an atom (we use the same letters for both) and each atom H produced by an argument is associated with a rule $H :- \mathbf{not}\ B_1, \ldots, \mathbf{not}\ B_m$**.**, where $B_1, \ldots, B_m$ are the atoms corresponding to arguments that attack the argument H.

To summarize, we have proposed an argumentation framework that puts conclusions in center stage, the CAFs. We have proved that they do guarantee nice correspondences with logic programs, going beyond AAFs — as CAFs do even display a correspondence between L-stable models and semi-stable conclusion labelings.

3.4 Finally: The Probability of an Argument

Having put in place an infrastructure for probabilistic argumentation, we are ready to ask: What exactly is the probability of an argument? We do not believe that there is a unique probability value that captures all uncertainty around a given argument, and we can explore here a few ideas by examining the structure of arguments.

Note, first, that there is a difference between the probability of a conclusion and of an argument. A conclusion may be accepted due to many different arguments operating with respect to distinct total choices.

Instead, when computing the "probability of an argument", we should take into account only the information attached to the argument. We need some machinery for that. Suppose we have a probabilistic logic program $\mathbf{P}$ and some argument A in that program. Suppose also that we run the WCG algorithm with $\mathbf{P}$ as input, but we treat probabilistic facts as if they were facts (this is illustrated by the right graph in Figure 2 in connection with Example 2). The set *rules* generated by the WCG algorithm is a set that contains rules, some of which may be facts, and probabilistic facts. We refer to this suitably enlarged set *rules* as the *probabilistic logic program attached to* A. We then propose:

Definition 6. *Given a probabilistic conclusion-augmented argumentation graph* $\mathbf{C}$ *generated from a probabilistic logic program* $\mathbf{P}$*, and an argument* A *in* $\mathbf{C}$*, the* probability of argument A *is the probability that the conclusion of* A *is* accepted *in the probabilistic logic program attached to* A*.*

An example is important here, so as to clarify the idea:

Example 5. *In Example 2, the probability of arguments* A_1 *and* A_3*, as well as the probability of arguments* A_2 *and* A_4*, is* 0.5*. The probability of arguments* A_5 *and* A_6 *is one; this makes sense because, if we view* **not** *as introducing assumptions in the argument, these assumptions should be taken to hold when looking at the argument in isolation. In fact, the atoms preceded by* **not** *behave as Toulmin's warrants — that is, as statements that should be accepted when evaluating an argument (Besnard and Hunter 2008). Finally, the probability of argument* A_7 *is zero; regardless of atom* A*, atom* E *cannot be* accepted*. Note that if we had simply erased atoms preceded by* **not***, we would obtain the rule* $E :- A.$ *and the probability of* E *would then be* 0.5*. Thus it makes sense to take the whole logic program inside an argument while evaluating the probability of the argument: the program may reveal internal connections between atoms.* □

Similar ideas have been proposed for variants of ABAFs, for instance by taking the probability of premises (Hunter 2013) or of premises and conclusion (Prakken 2018).

To conclude, we note that the conditional probability of the conclusion given the premises has been studied in the literature (Timmer et al. 2017). Such a notion deserves attention if we allow rules themselves to be probabilistically fired, for instance by resorting to ProbLog's *probabilistic rules* (Fierens et al. 2014). Such a rule is written $\alpha :: H :- \mathbf{B}.$, where $\mathbf{B}$ denotes the body; it is syntactic sugar for a rule $H :- \mathbf{B}.$ and a fresh probabilistic fact $\alpha :: F.$ whose atom F does not appear anywhere. For instance, suppose we have the following logic program associated with an argument:

$$H :- G_1, G_2, G_3, \mathbf{not}\ G_4., \quad 0.5 :: G_1., \quad 0.5 :: G_2.,$$
$$0.5 :: G_3 :- G_5, \mathbf{not}\ G_6. \qquad 0.5 :: G_5.$$

Then the probability of the argument is $(0.5)^4$, but the conditional probability of H given G_1, G_2, G_3, G_5 all accepted is 0.5. We might even condition on $\mathbf{not}\ G_4$, $\mathbf{not}\ G_6$, and this may be an avenue to pursue in the future. Here we take the position that the atoms preceded by **not** carry the warrants of the argument and thus must not be conditioned upon.

4 Conclusion

In this paper we have dealt with two interrelated issues: how to handle non-singleton/empty sets of stable models in probabilistic logic programming, and how to put together a practical and meaningful theory of probabilistic argumentation. These challenges are connected by the fact that probabilistic facts and rules can be used to build probabilistic arguments, and probabilistic argumentation has specific needs concerning absence of stable extensions. Our solution is to combine the credal semantics (for probabilistic facts) with the L-stable semantics (for logic programs).

We have introduced the L-credal semantics for logic programs. We derived the complexity of the inference problem in the propositional case, proving $\mathtt{PP}^{\Sigma_2^p}$-completeness. We conjecture that similar proofs can be used to prove $\mathtt{PP}^{\Sigma_3^p}$-completeness for the inference problem when predicates have bounded arity. This is our next step. Another future step is the development of inference algorithms, possibly by combining techniques for probabilistic and credal inference (Cozman and Mauá 2020; Fierens et al. 2014; Totis, Kimmig, and Raedt 2021).

We also contributed with a different route to Conclusion(Claim)-augmented Argumentation Frameworks, an extended version of Dung's AAFs that smoothly connect the semantics of logic programming and the semantics of argumentation. We obtained a connection between probabilistic logic programs and probabilistic argumentation that is important within the pipeline suggested by Figure 1.

Finally, we have proposed a definition for the probability of an argument. While a single concept does not seem to capture all the uncertainty carried by an argument, our proposal is a step forward that can be extended in future work.

Overall, we have addressed the conceptual foundation behind the pipeline in Figure 1; we know how to represent rules and probabilistic facts, and how to build, present and evaluate probabilistic arguments. Of course, we have captured only a fraction of logic programming and argumentation theory; much more work is due to capture various semantics, disjunctive heads and aggregates, supports, weights and preferences. We will focus on that in future work.

Acknowledgements

This work was carried out at the Center for Artificial Intelligence (C4AI-USP), with support by the São Paulo Research Foundation (FAPESP grant 2019/07665-4) and by the IBM Corporation. The first author is partially supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) grant 88887.616392/2021-00. The second author was partially supported by CNPq grant 312180/2018-7. We acknowledge support by CAPES - Finance Code 001.

References

Augustin, T.; Coolen, F. P. A.; de Cooman, G.; and Troffaes, M. C. M. 2014. *Introduction to Imprecise Probabilities*. Wiley.

Baral, C.; Gelfond, M.; and Rushton, N. 2009. Probabilistic reasoning with answer sets. *Theory and Practice of Logic Programming* 9(1):57–144.

Baroni, P.; Giacomin, M.; and Vicig, P. 2014. On rationality conditions for epistemic probabilities in abstract argumentation. In *Computational Models of Argument*, 121–132.

Besnard, P., and Hunter, A. 2008. *Elements of Argumentation*. MIT Press.

Bondarenko, A.; Dung, P.; Kowalski, R.; and Toni, F. 1997. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intellligence* 93:63–101.

Caminada, M., and Schulz, C. 2017. On the equivalence between assumption-based argumentation and logic programming. *Journal of Artificial Intelligence Research* 60:779–825.

Caminada, M.; Sá, S.; ao Alcântara, J.; and Dvorak, W. 2015. On the equivalence between logic programming semantics and argumentation semantics. *International Journal of Approximate Reasoning* 58:87–111.

Ceylan, Í. Í.; Lukasiewicz, T.; and Peñaloza, R. 2016. Complexity results for probabilistic Datalog$^{\pm}$. In *European Conference on Artificial Intelligence*, 1414–1422.

Cozman, F. G., and Mauá, D. D. 2017. On the semantics and complexity of probabilistic logic programs. *Journal of Artificial Intelligence Research* 60:221–262.

Cozman, F. G., and Mauá, D. D. 2020. The joy of Probabilistic Answer Set Programming: Semantics, complexity, expressivity, inference. *International Journal of Approximate Reasoning* 125:218–239.

Dung, P. M., and Thang, P. M. 2010. Towards (probabilistic) argumentation for jury-based dispute resolution. In *Frontiers in Artificial Intelligence and Applications*, volume 216, 171–182. IOS Press.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence* 77(2):321–357.

Dvorák, W., and Woltran, S. 2019. Complexity of abstract argumentation under a claim-centric view. *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019* 2801–2808.

Dvorák, W. 2012. *Computational Aspects of Abstract Argumentation*. Ph.D. Dissertation, Fakultät für Informatik der Technischen Universität Wien.

Dvořák, W.; Rapberger, A.; and Woltran, S. 2020. Argumentation semantics under a claim-centric view: Properties, expressiveness and relation to SETAFs. *17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020* 1:340–349.

Eiter, T.; Ianni, G.; and Krennwalner, T. 2009. Answer set programming: a primer. In *Reasoning Web: Semantic Technologies for Information Systems*.

Eiter, T.; Leone, N.; and Saccá, D. 1998. Expressive power and complexity of partial models for disjunctive deductive databases. *Theoretical Computer Science* 206:181–218.

Fierens, D.; Van den Broeck, G.; Renkens, J.; Shrerionov, D.; Gutmann, B.; Janssens, G.; and De Raedt, L. 2014. Inference and learning in probabilistic logic programs using weighted Boolean formulas. *Theory and Practice of Logic Programming* 15(3):358–401.

Hadjichristodoulou, S., and Warren, D. S. 2012. Probabilistic logic programming with well-founded negation. In *International Symposium on Multiple-Valued Logic*, 232–237.

Haenni, R. 2001. Cost-bounded argumentation. *International Journal of Approximate Reasoning* 26:101–127.

Haenni, R. 2009. Probabilistic argumentation. *Journal of Applied Logic* 7:155–176.

Heyninck, J., and Strasser, C. 2016. Relations between assumption-based approaches in nonmonotonic logic and formal argumentation. In *International Workshop on Non-Monotonic Reasoning*.

Heyninck, J. 2019. Relations between assumption-based approaches in nonmonotonic logic and formal argumentation: from structured argumentation to adaptive logics. *Journal of Applied Logics – IfCoLog Journal of Logics and their Applications* 6(2):317–357.

Howson, C., and Urbach, P. 1993. *Scientific Reasoning: the Bayesian Approach*. Chicago, Illinois: Open Court Publishing Company.

Hung, N. D. 2017. Inference procedures and engine for probabilistic argumentation. *International Journal of Approximate Reasoning* 90:163–191.

Hunter, A.; Polberg, S.; Potyka, N.; Rienstra, T.; and Thimm, M. 2021. Probabilistic argumentation: A survey. In *Handbook of Formal Argumentation*, volume 2. College Publications.

Hunter, A.; Polberg, S.; and Thimm, M. 2020. Epistemic graphs for representing and reasoning with positive and negative influences of arguments. *Artificial Intelligence* 281:103236.

Hunter, A. 2013. A probabilistic approach to modelling uncertain logical arguments. *International Journal of Approximate Reasoning* 54:47–81.

Li, H.; Oren, N.; and Norman, T. J. 2011. Probabilistic argumentation frameworks. In *International Workshop on Theories and Applications of Formal Argumentation*, 1–16.

Lukasiewicz, T. 2005. Probabilistic description logic programs. In *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2005)*, 737–749. Barcelona, Spain: Springer.

Mantadelis, T., and Bistarelli, S. 2020. Probabilistic abstract argumentation frameworks, a possible world view. *International Journal of Approximate Reasoning* 119:204–219.

Modgil, S., and Prakken, H. 2014. The *ASPIC*$^+$ framework for structured argumentation: a tutorial. *Argument and Computation* 5(1):31–62.

Molchanov, I. 2005. *Theory of Random Sets*. Springer.

Papadimitriou, C. H. 1994. *Computational Complexity*. Addison-Wesley Publishing.

Poole, D. 2003. First-order probabilistic inference. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 985–991.

Prakken, H. 2018. Probabilistic Strength of Arguments with Structure. In *International Conference on Principles of Knowledge Representation and Learning*, 158–167.

Rapberger, A. 2020. Defining argumentation semantics under a claim-centric view. In *CEUR Workshop Proceedings*, volume 2655. CEUR-WS.

Riveret, R.; Rotolo, A.; Sartor, G.; Prakken, H.; and Roth, B. 2007. Success chances in argument games: a probabilistic approach to legal disputes. In *Conference on Legal Knowledge and Information Systems*, 99–108.

Riveret, R.; Baroni, P.; Gao, Y.; Governatori, G.; Rotolo, A.; and Sartor, G. 2018. A labelling framework for probabilistic argumentation. *Annals of Mathematics and Artificial Intelligence* 83(1):21–71.

Sacca, D. 1997. The expressive powers of stable models for bound and unbound DATALOG queries. *J. Comput. System Sci.* 54(3):444–464.

Sato, T.; Kameya, Y.; and Zhou, N.-F. 2005. Generative modeling with failure in PRISM. In *International Joint Conference on Artificial Intelligence*, 847–852.

Sato, T. 1995. A statistical learning method for logic programs with distribution semantics. In *Conference on Logic Programming*, 715–729.

Thimm, M. 2012. A probabilistic semantics for abstract argumentation. In *European Conference on Artificial Intelligence (ECAI)*, 750–755.

Timmer, S. T.; Meyer, J.-J.; Prakken, H.; Renooij, S.; and Verheij, B. 2017. A two-phase method for extracting explanatory arguments from bayesian networks. *International Journal of Approximate Reasoning* 80:475–494.

Toni, F. 2014. A tutorial on assumption-based argumentation. *Argument and Computation* 5(1):89–117.

Totis, P.; Kimmig, A.; and Raedt, L. D. 2021. SMProbLog: Stable model semantics in ProbLog and its applications in argumentation. Technical Report arXiv:2110.01990, arXiv.

van Eemeren, F. H.; Garssen, B.; Krabbe, E. C. W.; Henkemans, A. F. S.; Verheij, B.; and Wagemans, J. H. M. 2014. *Handbook of Argumentation Theory*. Springer.

Čyras, K.; Heinrich, Q.; and Toni, F. 2021. Computational complexity of flat and generic assumption-based argumentation, with and without probabilities. *Artificial Intelligence* 293.

Wagner, K. W. 1986. The complexity of combinatorial problems with succinct input representation. *Acta Informatica* 23:325–356.

Walley, P. 1991. *Statistical Reasoning with Imprecise Probabilities*. London: Chapman and Hall.